16-833: Robot Localization and Mapping, Spring 2024

Homework 3 - Linear and Nonlinear SLAM Solvers

Jinjiang You (jinjiany)

Due: Friday Mar 29, 11:59pm, 2024

Your homework should be submitted as a **typeset PDF file** along with a folder including **code only (no data)**. The PDF and code must be submitted on **Gradescope**. If you have questions, post them on Piazza or come to office hours. Please do not post solutions or codes on Piazza. This homework must be done **individually**, and plagiarism will be taken seriously. You are free to discuss and troubleshoot with others, but the code and writeup must be your own. Note that you should list the name and Andrew ID of each student you have discussed with on the first page of your PDF file.

1 2D Linear SLAM

1.1 Measurement function (10 points)

$$h_o(\mathbf{r}^t, \mathbf{r}^{t+1}) = \mathbf{r}^{t+1} - \mathbf{r}^t = \begin{bmatrix} r_x^{t+1} - r_x^t \\ r_y^{t+1} - r_y^t \end{bmatrix}$$

$$H_o(\mathbf{r}^t, \mathbf{r}^{t+1}) = \begin{bmatrix} \frac{\partial h_{o,x}}{\partial r_x^t} & \frac{\partial h_{o,x}}{\partial r_y^t} & \frac{\partial h_{o,x}}{\partial r_x^{t+1}} & \frac{\partial h_{o,x}}{\partial r_y^{t+1}} \\ \frac{\partial h_{o,y}}{\partial r_x^t} & \frac{\partial h_{o,y}}{\partial r_y^t} & \frac{\partial h_{o,y}}{\partial r_x^{t+1}} & \frac{\partial h_{o,y}}{\partial r_y^{t+1}} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

$$h_l(\mathbf{r}^t, \mathbf{l}^k) = \mathbf{l}^k - \mathbf{r}^t = \begin{bmatrix} l_x^k - r_x^t \\ l_y^k - r_y^t \end{bmatrix}$$

$$H_l(\mathbf{r}^t, \mathbf{l}^k) = \begin{bmatrix} \frac{\partial h_{l,x}}{\partial r_x^t} & \frac{\partial h_{l,x}}{\partial r_y^t} & \frac{\partial h_{l,x}}{\partial l_x^k} & \frac{\partial h_{l,x}}{\partial l_y^k} \\ \frac{\partial h_{l,y}}{\partial r_x^t} & \frac{\partial h_{l,y}}{\partial r_y^t} & \frac{\partial h_{l,y}}{\partial l_x^k} & \frac{\partial h_{l,y}}{\partial l_y^k} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

1.2 Build a linear system (15 points)

See code.

1.3 Solvers (20 points)

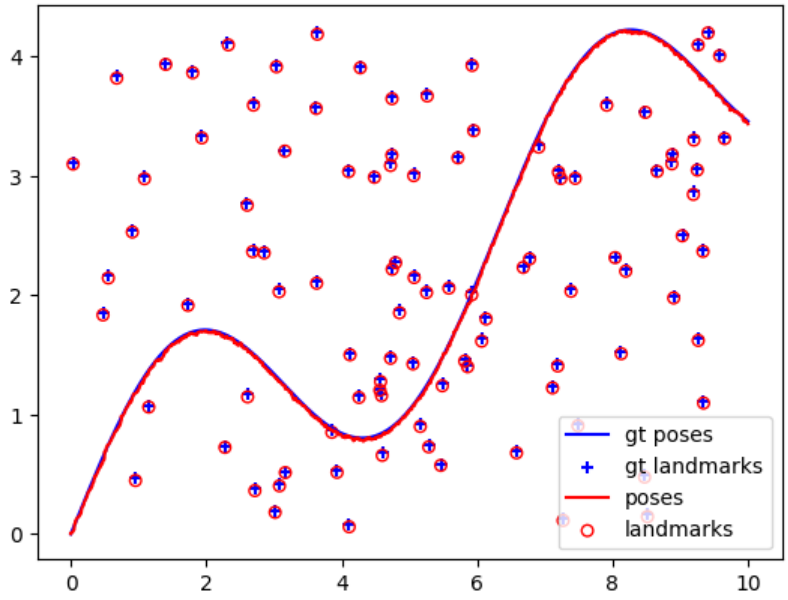
See code.

1.4 Exploit sparsity (30 points + 10 points)

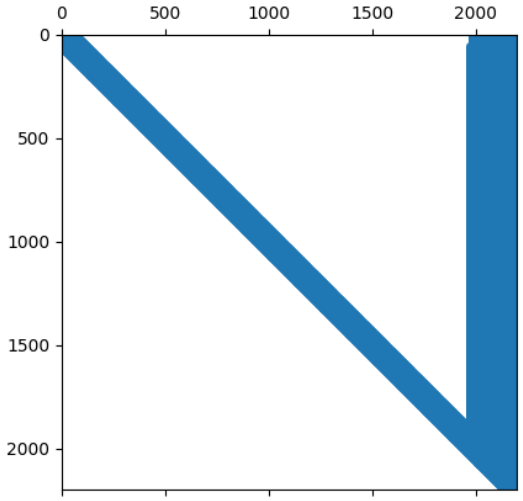
1.4.4 2d_linear.npz

Method	Average Time(s)
default	0.1037
pinv	2.6932
lu	0.0252
qr	0.4383
lu_colamd	0.1125
qr_colamd	0.3109

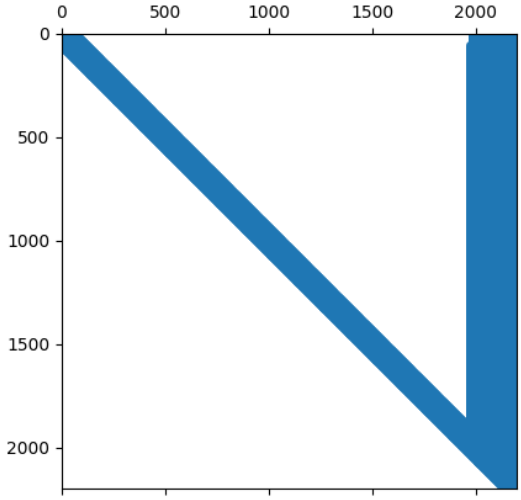
The trajectories of all the methods look the same, meaning that the implementations are correct:



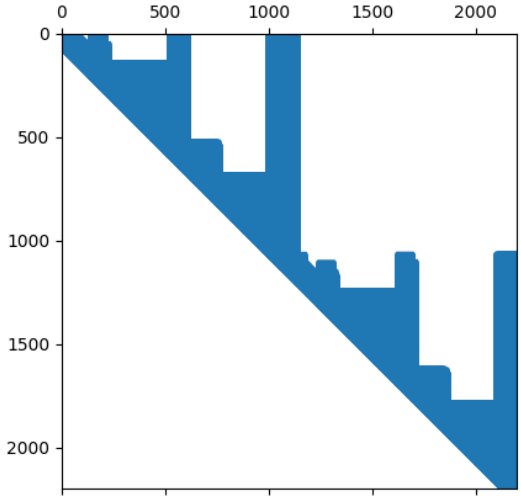
Below is the visualization of the matrices:
lu:



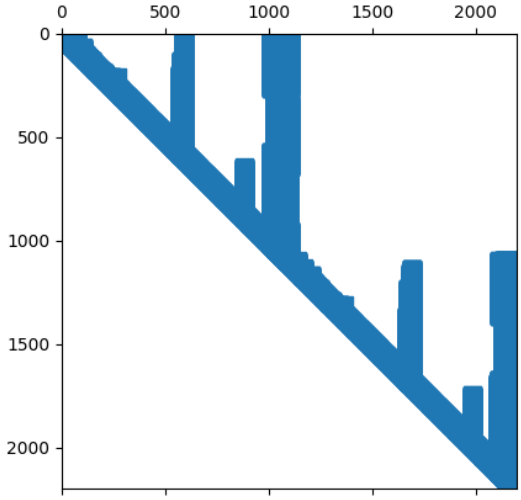
qr:



lu_colamd:



qr_colamd:



From the table we can see that LU (LU_COLAMD) is faster than QR (QR_COLAMD). Besides, QR_COLAMD is faster than QR. However, LU_COLAMD is slower than LU. The efficiency ranking is: LU >LU_COLAMD >QR_COLAMD >QR.

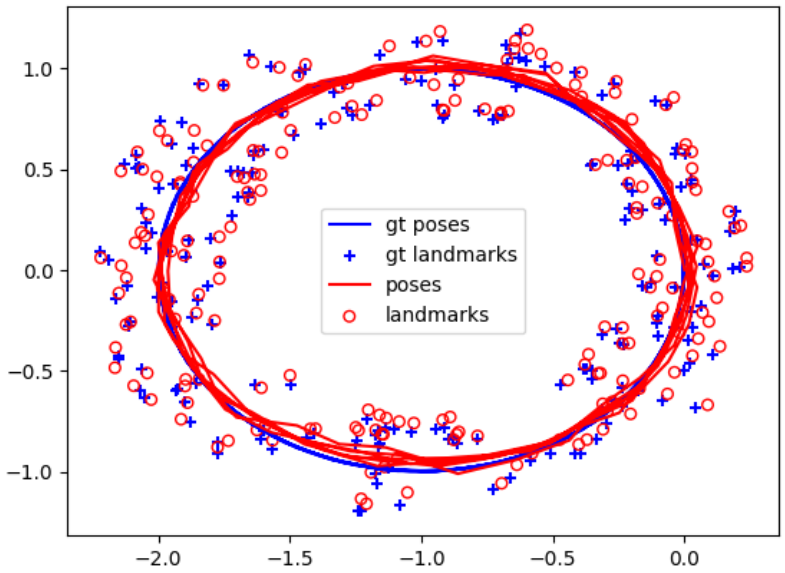
Compared with LU, QR is more numerically stable but slower, so it is reasonable that QR (QR_COLAMD) takes more time than LU (LU_COLAMD).

From the visualized figures we can see that the matrix of LU_COLAMD is not much sparser than that of LU. So this may be the reason for LU_COLAMD being slower than LU.

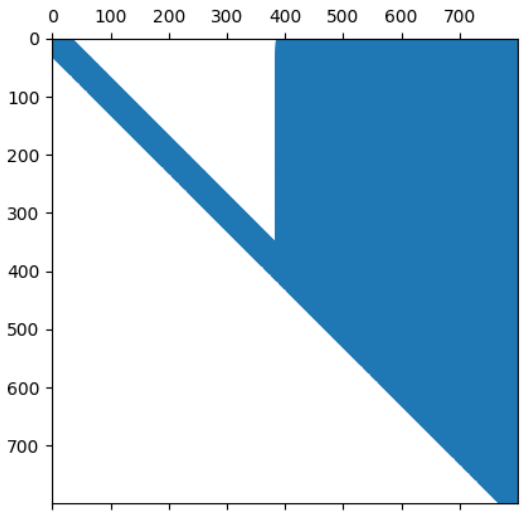
1.4.5 2d_linear_loop.npz

Method	Average Time(s)
default	0.0046
pinv	0.2760
lu	0.0178
qr	0.3047
lu_colamd	0.0049
qr_colamd	0.0226

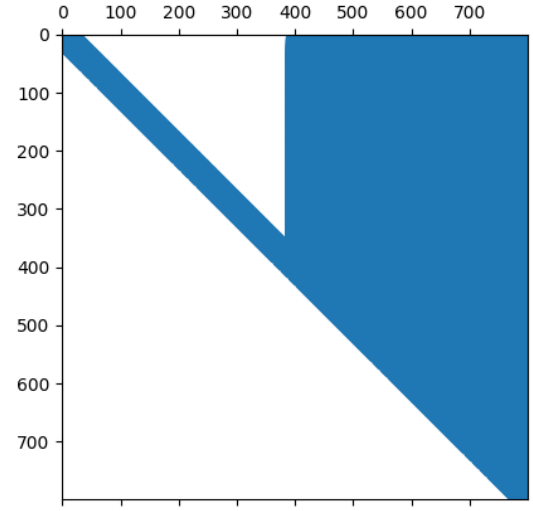
The trajectories of all the methods look the same, meaning that the implementations are correct:



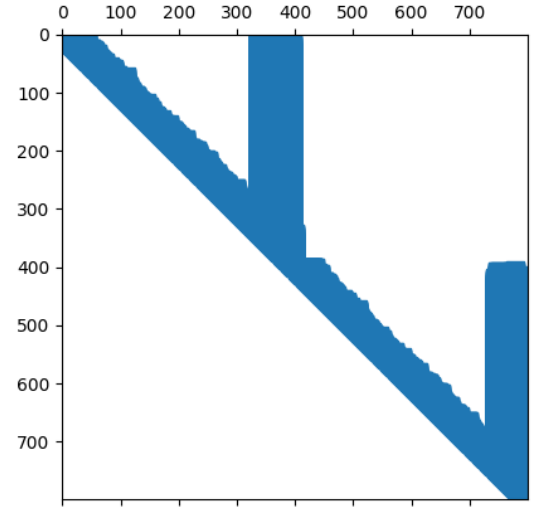
Below is the visualization of the matrices:
lu:



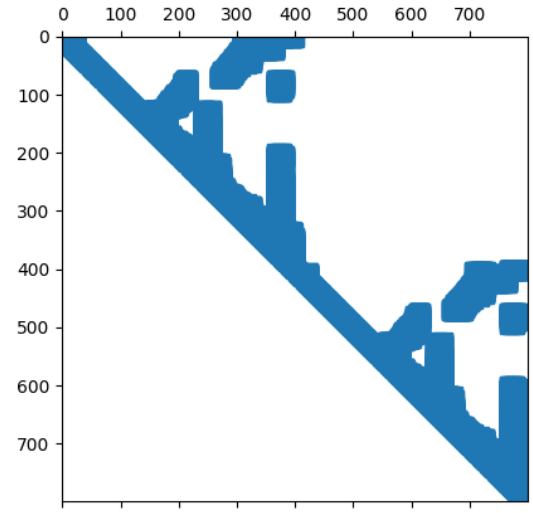
qr:



lu_colamd:



qr_colamd:



Similarly, LU (LU_COLAMD) is faster than QR (QR_COLAMD), for the same reason metioned above. The difference is that this time LU_COLAMD is faster than LU. The efficiency ranking is: LU_COLAMD >LU >QR_COLAMD >QR.

From the visualized figures we can see that the matrices of two reordered versions are much sparser than those of non-reordered versions. So the reordered versions are faster.

2 2D Nonlinear SLAM

2.1 Measurement function (10 points)

$$H_l(\mathbf{r}^t, \mathbf{l}^k) = \begin{bmatrix} \frac{\partial h_{l,x}}{\partial r_x^t} & \frac{\partial h_{l,x}}{\partial r_y^t} & \frac{\partial h_{l,x}}{\partial l_x^k} & \frac{\partial h_{l,x}}{\partial l_y^k} \\ \frac{\partial h_{l,y}}{\partial r_x^t} & \frac{\partial h_{l,y}}{\partial r_y^t} & \frac{\partial h_{l,y}}{\partial l_x^k} & \frac{\partial h_{l,y}}{\partial l_y^k} \end{bmatrix} = \begin{bmatrix} \frac{dy}{dy^2+dx^2} & \frac{-dx}{dy^2+dx^2} & \frac{-dy}{dy^2+dx^2} & \frac{dx}{dy^2+dx^2} \\ \frac{-dx}{\sqrt{dy^2+dx^2}} & \frac{-dy}{\sqrt{dy^2+dx^2}} & \frac{dx}{\sqrt{dy^2+dx^2}} & \frac{dy}{\sqrt{dy^2+dx^2}} \end{bmatrix}$$

where

$$dx = l_x^k - r_x^t$$

$$dy = l_y^k - r_y^t$$

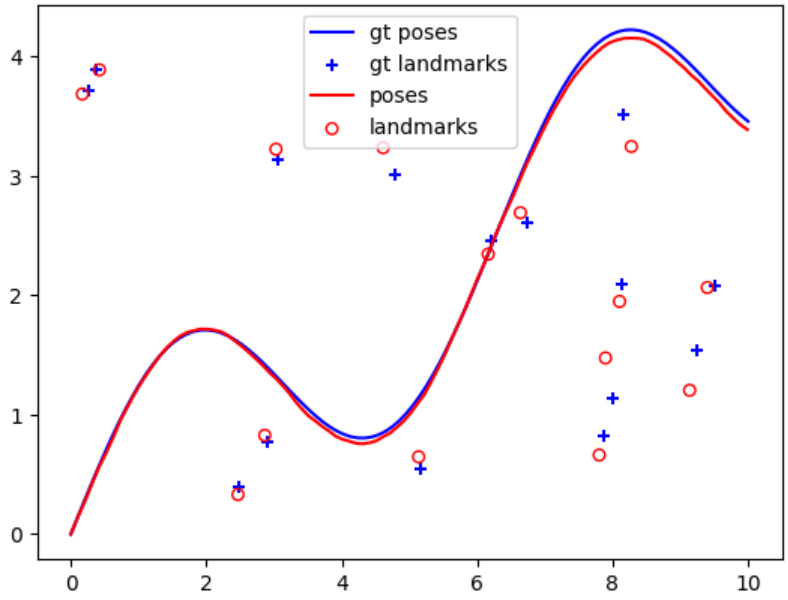
2.2 Build a linear system (15 points)

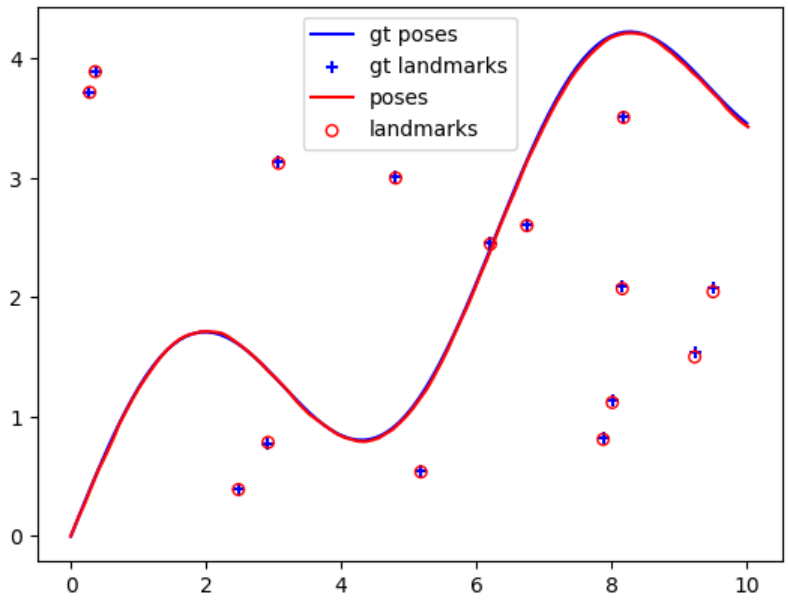
See code.

2.3 Solver (10 points)

Process `2d_nonlinear.npz`. Select one solver you have implemented, and visualize the trajectory and landmarks before and after optimization. Briefly summarize the differences between the optimization process of the linear and the non-linear SLAM problems. (10 points)

I have tried all the 6 methods and the results look the same. Below is the result of QR_COLAMD:





In the linear case, the solution can be got by directly solving a linear equation. However, in the nonlinear case, we need to solve the equation iteratively, and update our state vector incrementally.